Assignment 8 – From GANs to WGANs (In-Lab)

Weight: 4% (In-lab)
Deadline: 18-Nov-2025 (Tuesday)

Overview

In this in-lab exercise, you will explore the evolution of **Generative Adversarial Networks (GANs)** into their more stable variant, the **Wasserstein GAN (WGAN)**, using the **FashionMNIST** dataset.

You will first **implement the training loop for a vanilla GAN** to observe its instability in practice — why it often collapses — and then **implement WGAN improvements** to stabilize training and produce better samples.

Dataset & Environment

- **Dataset:** FashionMNIST (28×28 grayscale)
- **Device:** GPU (Colab or local)
- **Libraries:** PyTorch (torch / torchvision)
- **Notebook:** `A8.ipynb` (provided)

Part A — Implementing a Vanilla GAN (1%)

Goal: Implement the training loop for a simple GAN and observe its instability during training.

A.1 Tasks

1. **Implement the training loop for the Vanilla GAN**
    - Architecture is provided in the notebook.
    - Complete the forward and backward passes and optimization steps.
    - Use Binary Cross-Entropy (BCE) loss and a sigmoid-based discriminator.
2. **Train for 3 epochs, observe, and analyze plots**
    - Generator and Discriminator loss plots.
    - Sample images after each epoch.
    - Answer briefly (1–2 lines): What problems do you observe in the generated outputs or loss curves?

A.2 Expected Performance

- Training should complete successfully for a few epochs without errors or divergence.
- Loss values will fluctuate but remain finite.
- Generated outputs may begin to show coarse shapes or texture patterns. Clear object forms are not expected at this stage.
- Displaying the loss plot and a 4×4 sample grid is sufficient.

Part B — Implementing a WGAN (3%)

Goal: Define a new Critic and implement a training loop using the **Wasserstein loss** for smoother, more stable training.

B.1 Tasks

1. **Implement the Critic architecture**
   - Define a new network (similar to the provided Discriminator) without a final sigmoid activation.
   - Use LeakyReLU activations between layers.
2. **Implement the training loop using the Wasserstein loss**
   Use the following objectives:

$$\begin{aligned} L_D &= \mathbb{E}[D(x_{\text{real}})] - \mathbb{E}[D(x_{\text{fake}})] + \lambda \cdot \text{GP}, \\ L_G &= -\mathbb{E}[D(x_{\text{fake}})] \end{aligned} \tag{1}$$

   Implement a helper function to compute the **gradient penalty (GP)** using $\lambda = 10$.
3. **Use the Adam optimizer**
   Set learning rate and betas as follows:
   - $lr = 1 \times 10^{-4}$
   - $\beta_1 = 0.5$
   - $\beta_2 = 0.9$
4. **Train for 5 epochs**
   Run your implemented training loop and plot both the Critic and Generator losses after each epoch. Generate a 4×4 grid of 16 sample images from the trained generator.
5. **Answer briefly (1–2 lines):**
   How do the training behaviour and image quality differ from the vanilla GAN?

Deliverables Checklist

- **Notebook:** `A8.ipynb` (runs top-to-bottom on a clean runtime; includes implemented training loops for Parts A and B)
- **Report (PDF)** which includes:
  1. Plots of loss curves for both GAN and WGAN.
  2. Two image grids:
     (a) Vanilla GAN outputs (after 5 epochs).
     (b) WGAN outputs (after 5 epochs).
     (c) Short answers (A.3 and B.5).

NOTES:

- Focus on understanding *why* WGAN improves stability — not perfect visuals.
- Keep your networks small (≤ 4 layers per module) so training remains fast.
- All other components (dataset loading, plotting utilities, and model definitions) are provided in the starter notebook.

Marking Breakdown (4% total)

#	Requirement	Value (%)	Notes
1	Vanilla GAN training loop implemented and runs successfully	0.6	Training executes without errors and produces sample outputs
2	WGAN Critic and training loop implemented	1.6	Correct Critic architecture, Wasserstein losses, and gradient penalty implemented
3	Plotting and visualization of training progress	1.0	Includes loss curves for both GAN and WGAN, and 4×4 image grids for each
4	Short comparative answers or observations	0.8	Brief reflection on training behavior and stability improvements